20 | 模型部署：怎么发布训练好的机器学习模型？

2021-10-13 黄佳

《零基础实战机器学习》 课程介绍 >

讲述：黄佳

时长 20:31 大小 18.80M

你好，我是黄佳。欢迎来到零基础实战机器学习。

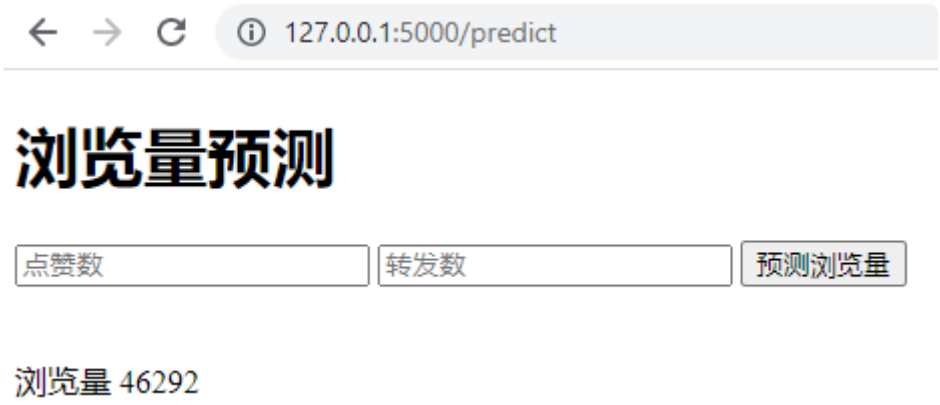
截止到现在，我们这个课程一直都是在本机上跑模型、分析和预测数据。但实际上，机器学习技术并不局限于离线应用，它也可以被部署到生产系统中发挥作用。

在具体部署时，出于对灵活性或者运行速度的需要，开发人员有可能会用 Java、C 或者 C++ 等语言重写模型。当然，重写模型是非常耗时耗力的，也不一定总是要这样做。像 Tensorflow、PyTorch、Sklearn 这些机器学习框架都会提供相应的工具，把数据科学家调试成功的机器学习模型直接发布为 Web 服务，或是部署到移动设备中。

在这一讲中，我们会通过一个非常简单的示例，来看看怎么把我们之前做过的预测微信文章浏览量的机器学习模型嵌入到 Web 应用中，让模型从数据中学习，同时实时接收用户

的输入，并返回预测结果。由于移动设备上的部署，还需要一些额外的步骤和工具，如 Tensorflow Lite 等，我这里不做展示，你有兴趣的话可以阅读相关文档。

我先给你看看搞定后的页面：



在进入具体的部署之前，我带你先整体了解一下，要搭建这个基于 Web 的机器学习应用，我们需要构建哪些模块。

整个项目需要构建哪些模块？

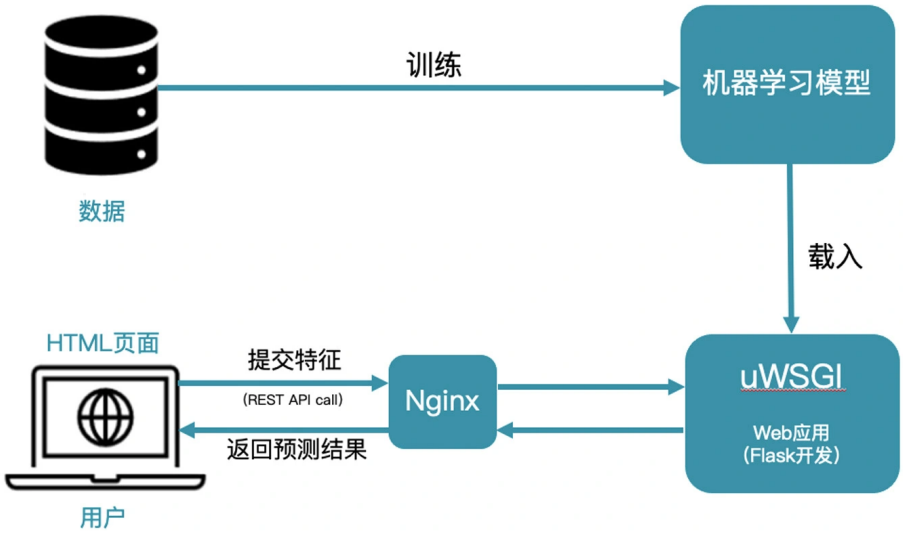
在这个项目中，我们实际上要构建的模块有三个：

机器学习模型：我们会把机器学习模型序列化，生成二进制文件，通过载入这个文件，Web 应用程序中就生成了可以调用的机器学习模型对象，我们可以根据 Web 端传入的点赞数、转发数，用模型来预测对应的浏览量。关于序列化，我们一会儿会详细讲到。

用 Flask 开发的 Web 应用：它通过 REST API，接收用户输入的点赞数和转发数信息，根据我们的模型计算预测值并返回它。Flask 是一个使用 Python 编写的轻量级 Web 应用框架，简单易用，这里我们将用它来开发 Web 应用。

HTML 页面模板：允许用户输入点赞数和转发数信息并提交给 Web 服务器，当 Web 应用收到信息之后，就会调用机器学习模型，并在 HTML 页面中显示浏览量的预测值。

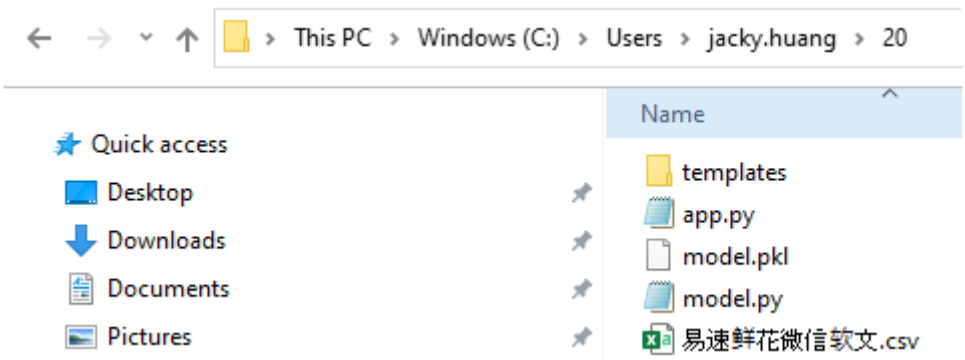
那么，这三个模块在整个 Web 应用中是如何协作的呢？我们来看看这张流程图：



首先，用户通过 HTML 页面输入点赞数和转发数，并发送预测的 REST API 请求。代理服务器 Nginx 接收到请求后，将请求转发给后端服务器 uWSGI。接着，后端服务器 uWSGI 会把请求继续转发给 Web 应用程序，这时候，Web 应用程序会载入机器学习模型，并根据接收到的点赞数和转发数进行预测。预测出的结果再经由 Web 应用程序、后端服务器 uWSGI、代理服务器 Nginx，一路返回给 HTML 页面，呈现给用户。

你可能已经注意到，在图的上半部分，还有一个“数据导入机器学习模型做训练”的过程。这个过程其实只需要完成一次，就是在我们首次创建好模型的时候，用提前准备好的数据集训练模型。后续 Web 应用程序每次载入模型都不需要再调用数据集进行训练。

基于上面这个流程，我们再来看下这个项目的目录结构（你可以在这里下载这些文件）：



可以看到，在这个项目中，包含了 5 个文件（文件夹），分别是：

templates：这个文件夹中存放着 HTML 文件 **index.html**，这是给用户看的网页模板，也就是对应着用户能够浏览并输入信息的 HTML 页面；

app.py：它就是 Web API 的开发源文件，负责部署机器学习模型；

model.py：这是一个 Python 脚本文件，存放着我们创建、拟合机器学习模型的代码。Python 脚本文件不是像 Jupyter Notebook 那样一个单元格，一个单元格交互地执行，而是在命令行整体执行。这个文件执行后会生成一个存储模型的二进制文件 model.pkl，提供给 Web API 导入。

model.pkl：就是刚才讲的 model.py 执行时生成的文件，用来存储模型。

易速鲜花微信软文.csv：这就是我们用来训练模型的数据集。

下面我们就从最简单、最容易理解的 HTML 页面开始，一步步搭建来这个项目。

创建 HTML 模板

因为我们现在不再是从本机上跑机器学习模型，那么就需要有一个 HTML 页面，提供给用户通过网络来输入要预测的东西（也就是特征）。这也就是我们通常说的前端，这个不难理解。

HTML 的语法也很简单，具体代码如下：

复制代码

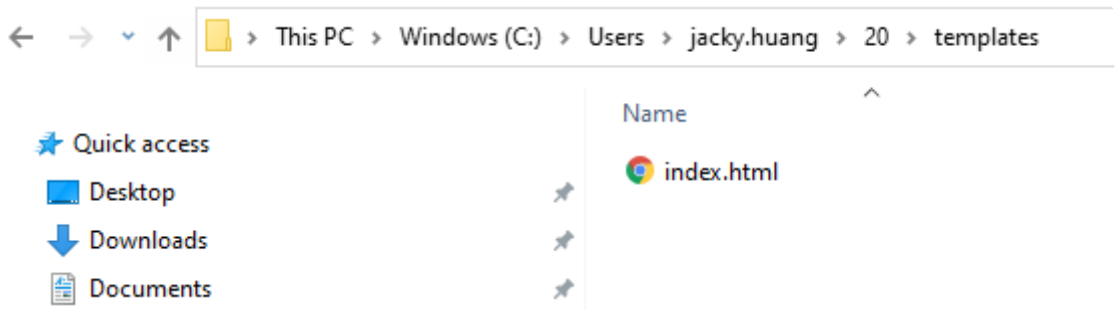
```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>机器学习部署应用</title>


</head>


 <div class="login">
  <h1>浏览量预测</h1>


      <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
        <input type="text" name="点赞数" placeholder="点赞数" required="required
    <input type="text" name="转发数" placeholder="转发数" required="required" />
        <button type="submit" class="btn btn-primary btn-block btn-large">预测
```

```
    </form>


    <br>
    <br>
    {{ prediction_text }}


  </div>
</body>
</html>
```

这段代码其实构建了这样一个页面：

在这个页面中，用户需要填写两个字段：一个是点赞数，另一个是转发数。这二者都是机器学习模型的特征字段。在这两个字段的旁边还有一个 Button“预测浏览量”，用户输入好特征字段后，就可以直接点击这个 Button，此时页面会发送 Request 给 Web 应用，让它返回模型所预测的标签。

现在，我们把上面的 HTML 代码 Copy 下来，另存为 HTML 格式的文档，并放在项目结构的 templates 目录之下就好了。注意，它目前只是一个模板，其中的具体内容还要通过 Web 应用进行动态的渲染，这个我们等会再细说。

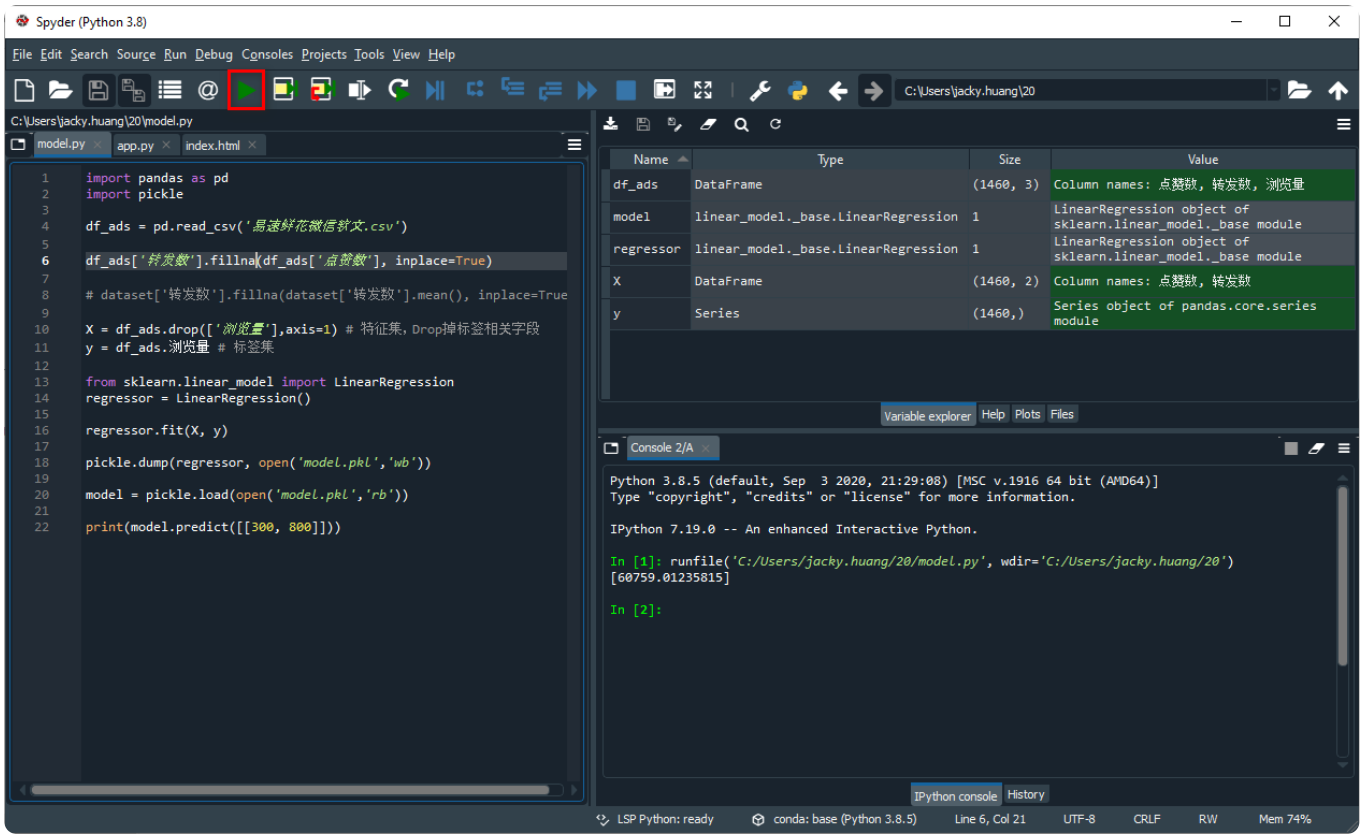


网页模板创建好了之后，我们就要创建能够预测微信软文浏览量的机器学习模型。这个机器学习的创建流程非常简单，你可以回忆一下第 3 讲和第 4 讲的内容，同时，你也可以在这里找到“易速鲜花微信软文”数据集。下面，我会带你重新创建一个极简的模型，你可以把它视为我们对之前内容的一次复习。

创建机器学习模型

整个创建过程说起来很简单。首先，我们使用 Pandas 的 fillna 函数，处理一下数据集中点赞数字段的缺失值（你也可以考虑用均值填充）。然后，我们再创建线性回归模型作为机器学习算法，并进行拟合和预测。最后，我们把这个模型保存为 model.py 文档。

这里你需要注意的是，“.py”格式的 Python 脚本文件可以整体运行，它不是像 Jupyter Notebook 中那样按每个单元格来一步步运行，这其实只是为了简化流程。而编辑器，我使用的是 Annaconda 中自带的 Spyder（如下图所示），当然你也可以使用其他 Python 开发环境来编辑，这并没有特别的限制。



Spyder中的机器学习模型代码

我们看一下整个创建机器学习模型的 Python 代码：

复制代码

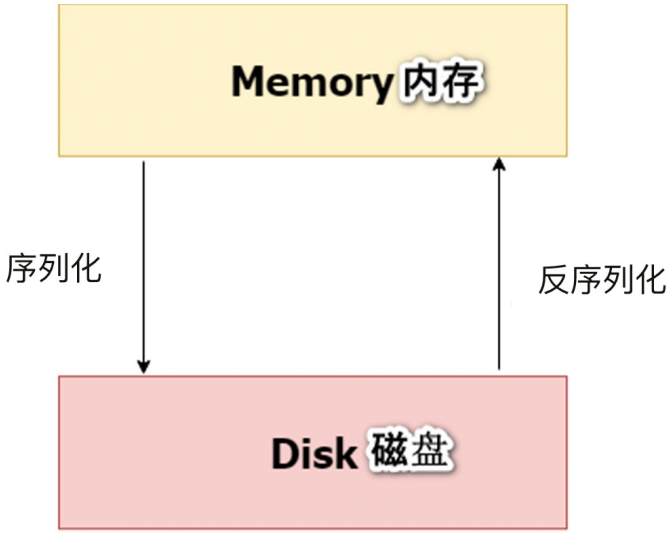
```
import pandas as pd # 导入Pandas
import pickle # 导入序列化工具Pickle


df_ads = pd.read_csv('易速鲜花微信软文.csv') #导入数据集


df_ads['转发数'].fillna(df_ads['点赞数'], inplace=True) #一种补值方法


X = df_ads.drop(['浏览量'],axis=1) # 特征集，Drop掉标签相关字段
y = df_ads.浏览量 # 标签集


from sklearn.linear_model import LinearRegression #导入线性回归模型
regressor = LinearRegression() #创建线性回归模型


regressor.fit(X, y) #拟合模型


pickle.dump(regressor, open('model.pkl','wb')) #序列化模型（就是存盘）


model = pickle.load(open('model.pkl','rb')) #反序列化模型（就是再导入模型）


print(model.predict([[300, 800]])) #进行一个预测
```

这段 Python 代码就是我们反复操练过的机器学习实战 5 步，相信你并不陌生。代码中唯一的新东西是 Pickle。Pickle 是 Python 中的序列化和反序列化工具，可以将 Python 对象以文件的形式存放在磁盘上，以供后面的 Web 应用调用它。

具体来讲，Pickle 会把内存中的模型对象（也就是 Python 对象），转换为 0 和 1 的字节流，形成字节流文件 model.pkl 存储到磁盘，这个过程叫存盘，也叫“序列化”。

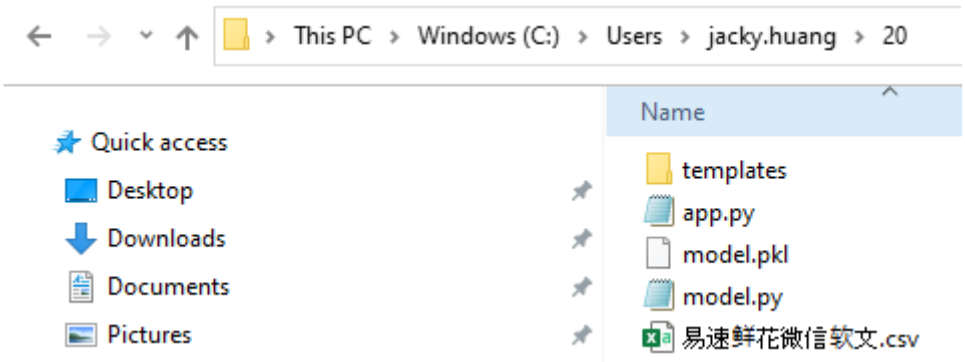
这个字节流文件 model.pkl 可以被保存到磁盘的任意地方，当我们需要调用模型的时候，Pickle 会把磁盘中的 model.pkl 再转换为 Python 对象，存在内存中。这个时候模型就被还原了，可以使用了。这个过程就叫做“反序列化”。

在上面的代码中，pickle.dump() 做的就是序列化，pickle.load() 做的就是反序列化。对于这两个过程，你可以通过下面这张图做进一步的理解：



序列化和反序列化

当我们把上面那段 Python 代码存为 model.py 文档，并在 Spyder 中运行 model.py 后，模型就完成了创建、训练和拟合的过程，同时也经历了序列化。这时候，model.py 所在目录下就出现了一个字节流文件 model.pkl 文件：



我们前面提到，在 Web 应用调用模型的过程中，需要把模型对象做一个转换，然后模型才能被使用。虽然我们用 Pickle 实现这个过程比较方便，但是它要求应用程序也是 Python 语言开发的，否则应用程序无法读取 Pickle 所生成的序列化文件。

显然，这并不适用于我们所有的应用开发需求。因此，这里我再给你介绍几种模型序列化和导出的格式，如果你在工程实践中需要使用，参考相关文档即可，它们的原理和 Pickle 是类似的。

Joblib：Joblib 是一组在 Python 中提供轻量级流水线（pipelining）的工具，它的功能和 Pickle 相似。Joblib 通过缓存机制将磁盘上的文件隐式链接到原始 Python 对象

的执行上下文，并支持简单的并行计算。和 Pickle 相比，它的优势是并行化、良好的日志、追踪功能以及大数据支持。

ONNX：ONNX 是一种开放神经网络交换格式，支持跨语言存储和移植预测模型。大多数深度学习库都支持它，其中，sklearn 有专门的扩展库，负责把模型转换为 ONNX 的格式。

PMML：PMML 是预测模型的另一种交换格式。与 ONNX 一样，sklearn 也有一个专门的扩展库用于将模型转换为 PMML 格式。不过，它的缺点是仅支持某些类型的预测模型。 PMML 历史悠久，出现于 1997 年，因此，有大量的应用程序使用这个格式，例如 PEGA 和 SAP 等。

POJO 和 MOJO：POJO 和 MOJO 是H2O.ai的导出格式，旨在为 Java 应用程序提供一个易于嵌入的模型。然而，它们仅适用于使用 H2O 的平台。

好了，现在我们有了 HTML 网页模板和机器学习模型，下一步，我们开始创建 Web 应用，用这个应用来调用机器学习模型。

创建 Web 应用，并在其中调用模型

创建 Web 应用并不复杂，我们用 Flask 就可以轻松完成。Flask 是一个使用 Python 编写的轻量级 Web 应用框架，它内置开发服务器和调试器，也支持 RESTful 请求，不需要我们做复杂的搭建。

我们前面说，这个 Web 应用的主要作用是，在接收到 HTML 页面传来的信息后，调用模型计算出预测结果，然后把预测结果返回给 HTML 页面。

当然，这只是一个大致的过程，详细来讲，这里面有两个关键步骤需要你注意。第一，在调用模型的时候，我们需要把模型从字节流文件反序列化为 Python 对象，然后这个模型才能正式被使用，这一点我们在前面提到过。

第二，Web 应用在接收到用户输入的信息后，实际上会调用 index.html 模板生成一个 /predict 页面，这个页面是最终呈现给用户的，其中会包含预测结果。在模型预测出结果后，Web 应用会把这个结果提交到这个 /predict 页面，返回给用户。

理解了这两点后，我们现在来看具体的代码：

复制代码

```
import numpy as np #导入NumPy
from flask import Flask, request, render_template #导入Flask相关包
import pickle #导入模块序列化包


app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb')) #反序列化模型


@app.route('/')
def home(): # 默认启动页面
    return render_template('index.html') # 启动index.html


@app.route('/predict',methods=['POST'])
def predict(): # 启动预测页面


    features = [int(x) for x in request.form.values()] # 输入特征
    label = [np.array(features)] # 标签
    prediction = model.predict(label) # 预测结果


    output = round(prediction[0], 2) #输出预测结果


    return render_template('index.html', #预测浏览量
                           prediction_text='浏览量 $ {}'.format(output)


if __name__ == "__main__": # 启动程序
    app.run(debug=True)
```

那么这个 Web 应用，就把我们之前训练好的模型和 HTML 页面串联起来了，并能接收 HTML 请求，返回结果。

考虑到你可能不太理解这些代码的具体含义，现在我给你简单讲解一下。前几行是导入相关包，这没什么好解释的，我们直接从第 6 行代码开始看。第 6 行 model = pickle.load(open(‘model.pkl’, ‘rb’)) 的意思就是导入序列化后的模型，并进行反序列化。

紧接着下一行代码 @app.route(‘/index’) 是一个装饰器。简单来说，它只是将它下面定义的方法映射到装饰器中提到的 URL，也就是每当用户访问该 URL ‘/’ （完整地址也

会有一个 ip 地址和一个端口号，比如 http://127.0.0.1:5000/ ），index() 方法将被自动调用， index() 方法返回 index.html 。注意，我们在 index.html 为用户提供了两个输入框，用户可以在其中输入任意的点赞数和转发数的值。

而在 home() 函数中，flask.render_template() 会查找 index.html 模板，并渲染出（也就是在运行时动态生成）一个最终给到用户的 HTML 页面。

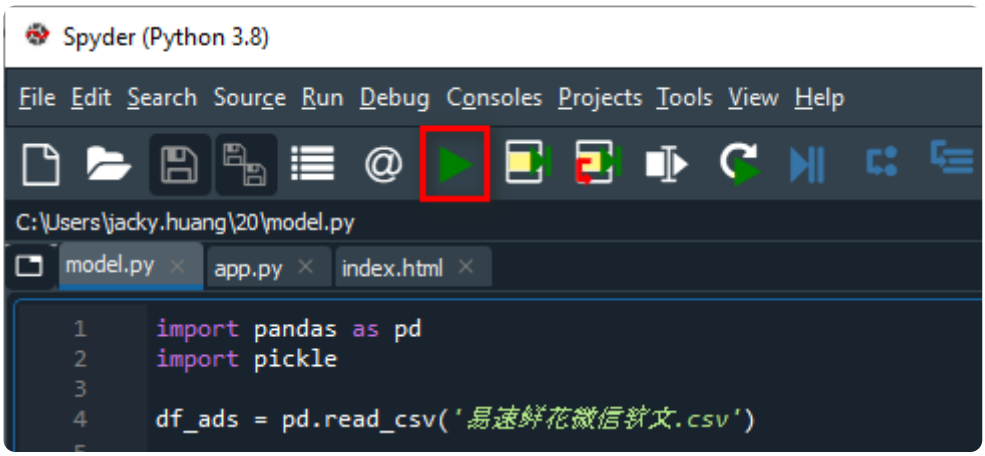
第 12 行代码 @app.route（‘/predict’）也是一个装饰器，这个装饰器将 predict() 方法映射到以“/predict”结尾的 URL ，它接受用户提供的输入，完成所有预处理，生成最终特征集，然后调用模型并获得最终预测值。

而其中 render_template() 函数所做的工作就是动态渲染 HTML 页面部分，将预测结果放入 HTML 页面中。只有当用户输入特征，并点击 index.html 页面中的提交按钮时，预测结果才会被显示。在 index.html 模板中，我们已经为这些变量值创建了占位符，而 render_template 负责获取模板文件夹中的 index.html，把预测值放进 HTML 模板，渲染出包含最终预测结果的 HTML 页面。

通过上述讲解，我想你应该可以理解这段创建 Web 应用的代码了。那到这里呢，我们所有的编码工作也就完成了！下面，我们就来启动这个 Web 应用。

使用已经发布的 App

我们单击 Spyder 的 Run 按钮就能启动 Web 应用。当然，你还可以在命令行界面中使用 python app.py 来运行它。

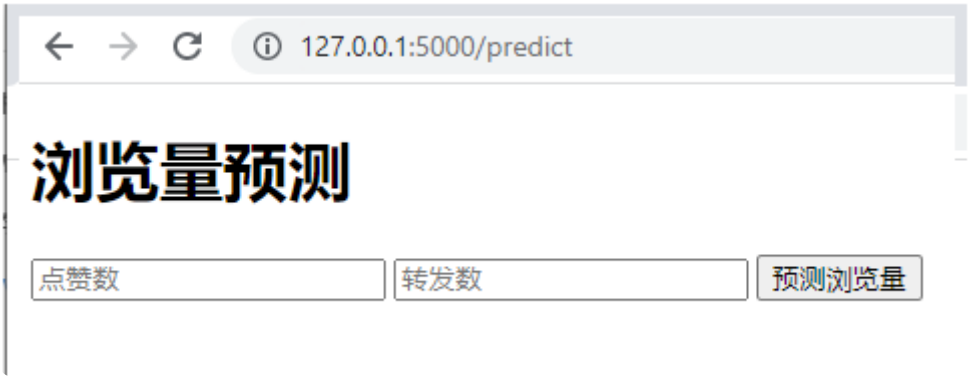


App 启动运行后，IPython Console 中的输出如下：

```
runfile('C:/Users/jacky.huang/20/app.py', wdir='C:/Users/jacky.huang/20')
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deploy
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
```

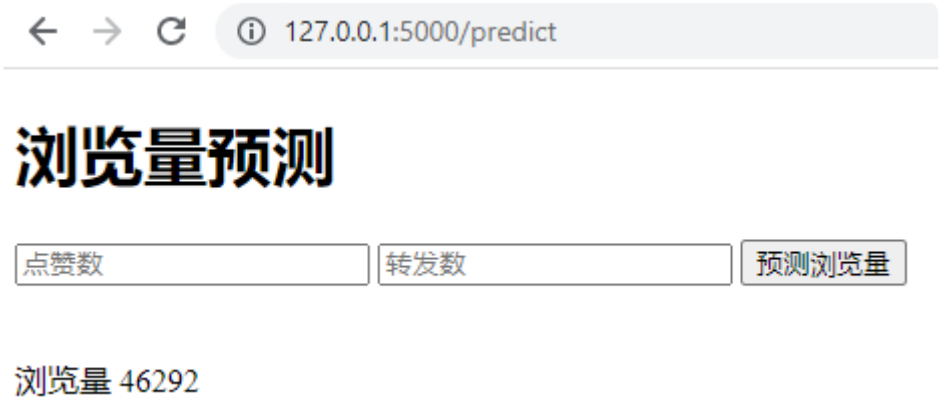
这表明，此时 Web 应用已经开始在 WSGI 服务器上运行，等候客户端页面随时发送请求。

现在，我们打开一个本机浏览器，访问 http://127.0.0.1:5000/ 页面。服务器返回如下 HTML 页面：



然后，我们充当用户角色，输入点赞数和转发数，希望基于这些数值来预测浏览量：

点击“预测浏览量”提交数据后，我们得到了浏览量预测结果：



这说明服务器成功接到 HTML 页面发送过来的 POST API 的请求，并根据模型的预测，通过 http://127.0.0.1:5000/predict 页面返回了预测结果：当点赞数为 500，转发数为 300 时，机器学习模型预测的文章浏览量是 46292。

这样，我们就把一个简单的机器学习模型打包成了一个 Web 应用，成功的部署到网页中啦！

总结一下

到这里，我们这一讲就结束了。现在，我们整体做个回顾。

通常，我们会把机器学习模型部署到生产系统（Web 端或移动端）中投入使用。部署机器学习模型有多种方式，比如开发人员重新编写模型、通过 Web 应用把模型部署到 Web 服务器、通过 Tensorflow Lite 等工具部署到移动端等等。

这一讲我们重点介绍了怎么把机器学习模型发布为 Web 应用。总体上，有这么几个开发环节：

1. 开发前端网页 HTML 呈现模板，用于传递用户输入的特征，同时显示服务器返回的预测结果；
2. 创建机器学习模型，对于模型的训练数据集，我这里仍然使用了 csv 文件；
3. 通过 Pickle 工具进行序列化，保存机器学习模型；
4. 用 Flask 框架开发一个轻量级的 Web 应用。其中，我们是先反序列化了机器学习模型，然后再把机器学习应用部署到 Web 服务器上。

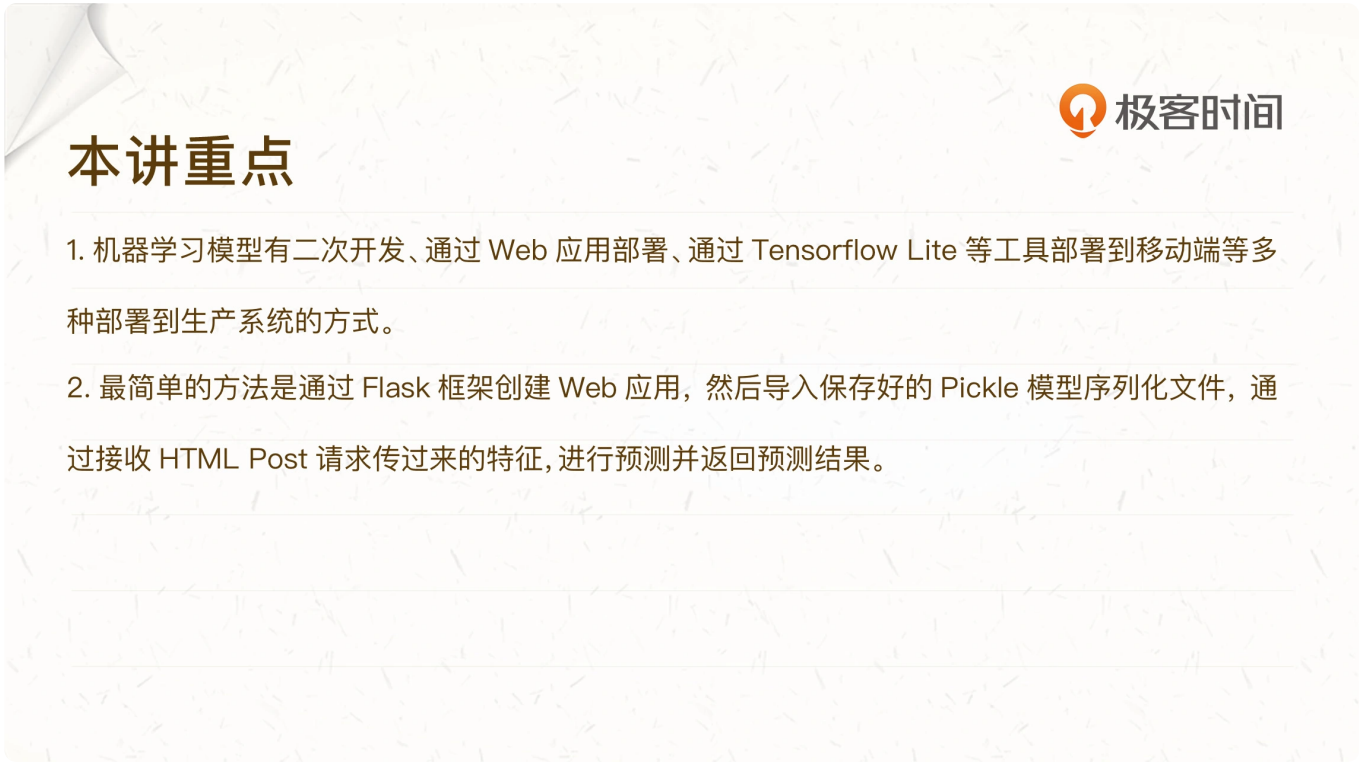
实际上，这几个环节都分别有多种实现方式，比如在模型训练数据时，你可以通过访问数据库的方式来获取实时的数据文件；在模型序列化时，你也可以使用 Joblib 来代替 Pickle；在开发 Web 应用时，你可以选择除 Flask 之外的其他框架……具体方式，你可以根据实际需要灵活选择。

思考题

下面，我给你留两道思考题。

1. 这里我是把预测微信软文浏览量的模型集成在了 Web 应用中，你是否可以选择其它的实战模型（比如预测用户的 LTV 价值），也把它集成在 Web 应用中，通过 HTML 访问并预测。
2. 你还有哪些机器学习模型上线部署的经验，可否跟我分享一下？

欢迎你在留言区和我分享你的观点，如果你认为这节课的内容有收获，也欢迎把它分享给你的朋友，我们下一讲再见！


极客时间
本讲重点
1. 机器学习模型有二次开发、通过 Web 应用部署、通过 Tensorflow Lite 等工具部署到移动端等多种部署到生产系统的方式。
2. 最简单的方法是通过 Flask 框架创建 Web 应用，然后导入保存好的 Pickle 模型序列化文件，通过接收 HTML Post 请求传过来的特征，进行预测并返回预测结果。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 1 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 19 | 胸有成竹：如何快速定位合适的机器学习算法？

下一篇 21 | 持续精进：如何在机器学习领域中找准前进的方向？

精选留言 (1)

写留言

高绥凯
2021-10-13

自己运行app.py函数的时候，经常会出现报错，不知道是哪里错误了？
runfile('D:/PycharmProjects/FirstWeb/app.py', wdir='D:/PycharmProjects/FirstWeb')
Traceback (most recent call last):

File "<ipython-input-33-aedc2d900ec7>", line 1, in <module>…

展开

作者回复: UnsupportedOperation: not writable，通常是在写入只读文件的时候报错。看一看是否对所有文件有写权限，是不是有一些文件是只读文件。这只是猜测，我没有遇到过同样错误。
我们看看其它同学是否遇到了类似的问题呢？

1